Neural Comput & Applic (2014) 24:723–734
DOI 10.1007/s00521-012-1285-7

ORIGINAL ARTICLE

Self-adaptive constrained artificial bee colony for constrained numerical optimization

Xiangtao Li · Minghao Yin

Received: 2 July 2012 / Accepted: 17 November 2012 / Published online: 6 December 2012

Abstract The artificial bee colony is a simple and effective global optimization algorithm. It has been successfully applied to solve a wide range of real-world optimization problem, and later, it was extended to constrained design problems as well. This paper describes a self-adaptive constrained artificial bee colony algorithm for constrained optimization problem based on feasible rule method and multiobjective optimization method. The employed bee colony severs as the global search engine for each population based on feasible rule. Then, the onlooker bee colony can explore the new search space based on the multiobjective optimization. In order to enhance the convergence rate of the proposed algorithm, a self-adaptive modification rate is proposed to make the algorithm can change many parameters. To verify the performance of our approach, 24 well-known constrained problems from 2006 IEEE congress on Evolution Computation (CEC2006) are employed. Experimental results indicate that the proposed algorithm performs better than, or at least comparable to, state-of-the-art approaches in terms of the quality of the resulting solutions from literature.

Keywords Artificial bee colony · Multiobjective optimization · Feasible rule · Improved algorithm · Exploration · Exploitation

X. Li · M. Yin (✉)
School of Computer Science and Information Technology, Northeast Normal University, Changchun 130117, China
e-mail: Minghao.Yin1@gmail.com

1 Introduction

Constraint optimization problems play an important role in many science and engineering disciplines. The constrained optimization problem can be formulated as the nonlinear programming is to find x so as to

$$\min_x f(x)$$

where $x \in F \subseteq S$ and $f(x)$ is the objective function. $F \in S$ is the feasible region, and S is the decision space bounded by the lower and upper bounds:

$$L_i \le x_i \le U_i, \quad 1 \le i \le n,$$

whereas $F \in S$ is the feasible region defined by a set of m additional linear and nonlinear constraints:

$$\begin{aligned} g_j(x) &\le 0, \quad j = 1, \ldots, q \\ h_j(x) &= 0, \quad j = q+1, q+2, \ldots, m \end{aligned}$$

where $g_j(x)$ is the jth inequality constraint and $h_j(x)$ is the jth equality constraint. In both cases, constraints could be linear or nonlinear. The degree of constraint violation of individual x on the jth constraint is calculated as follows:

$$G_j(x) = \begin{cases} \max(0, g_j(x)) & 1 \le j \le q \\ \max\left(0, \left|h_j(x) - \varepsilon\right|\right) & q+1 \le j \le m \end{cases} \quad (1)$$

where ε is a positive tolerance value for equality constraints. Then, $G(x) = \sum_{j=1}^{m} G_j(x)$ reflects the degree of constraint violation of the individual x.

During the past decade, we have viewed different kinds of meta-heuristic algorithms advanced to handle constraint optimization problems (COPs) [1–3]. Among them, meta-heuristic-based methods, such as genetic algorithm (GA), particle swarm optimization algorithm (PSO), estimation of distribution algorithms (EDA), ant colony optimization

(ACO), simulated annealing (SA), biogeography-based optimization (BBO), differential evolution (DE), and artificial bee colony (ABC) [4–12], may be one of the most popular methods. However, meta-heuristic algorithms always perform an unconstrained search, Therefore, while using EAs to solve constrained optimization problem, constrained handle methods need to add into the algorithm. Different constrained handle methods include penalty function, feasible over infeasible solutions, and multiobjective optimization method. Penalty function is the most common constraint handling techniques due to their simplicity and ease of implementation. The crucial idea converts the constrain optimization into unconstrained optimization problem. For feasible over infeasible solution, this method shows the feasible is better than the infeasible solution. For multiobjective optimization methods, the crucial idea of this method is to convert constraint optimization problem into unconstrained multiobjective optimization problems. Then, multiobjective optimization methods are used to handle the converted problem [13, 14]. In this paper, we will use the feasible rule and multiobjective optimization as the constraint handling method.

Artificial bee colony algorithm [12] (ABC), is a population-based heuristic evolutionary algorithm inspired by the intelligent foraging behavior of the honeybee swarm. In ABC, a honeybee colony consists of three kinds of bees: employed bee, onlooker bees, and scout bees. Employed bees are responsible for exploiting the nectar sources explored before, sharing their information with onlookers within the hive. Onlooker bees wait in the hive and decide on a food source to exploit based on the information shared by the employed bee colony. Scout bees choose one of the most inactive solutions and then replace it by a new randomly generated solution. Numerical comparisons show that the performance of the ABC algorithm is better than other swarm intelligent algorithms with the advantage of employing fewer control parameters. ABC is easy to implement and has been proven to perform well on many practical optimization problem [15–17]. Since the success of ABC in many unconstrained optimization problem, ABC has attracted many researchers to extend it to solve the constrained optimization. Mezura-Montes et al. [18] propose a modified artificial bee colony algorithm to solve constrained numerical optimization problems, some modifications related with the selection mechanism, the scout bee operator, and the equality and boundary constraints are made to the algorithm with the aim to modify the original algorithm. The results obtained in the experiment showed that M-ABC is better than the original ABC. Mezura-Montes et al. [19] propose a novel artificial bee colony algorithm that enhances the three kinds of bee colony and uses a dynamic tolerance control method to make the approach to find the feasible solution. Karaboga and Akay [20] propose a modified ABC algorithm for constrained optimization problems. For constraint handling, ABC algorithm uses Deb's rules, and a probabilistic selection scheme is proposed between feasible solutions and infeasible solutions. Akay and Karaboga [21] use ABC algorithm to solve large-scale optimization problems and engineering design problems. For the constrained engineering problem, the basic ABC algorithm was extended simply by adding a constraint handling technique into the selection step of the ABC algorithm. Brajevic and Tuba [22] propose an upgraded artificial bee colony for constrained engineering problem. The algorithm improved fine-tuning characteristics of the modified rate parameter and employs modified scout bee part.

Recently, Wang et al. [23] propose a hybrid multiswarm particle swarm optimization for constrained optimization. At each generation, the swarm is split into several subswarms to taking advantage of PSO as the search algorithm. Then the differential evolution is as the new update method to update the personal best solution [23]. For the constraint handling method, the feasibility rule was used to choose particles in the population. Inspired by [23], we propose self-adaptive constrained artificial bee colony (SACABC) based on feasible rule and multiobjective optimization problem. The employed bee colony uses the new search method based on the feasible rule to generate the offspring. Then, the onlooker bee colony is used to explore the new search space using the new search method based on multiobjective optimization problem. Pareto dominance used in multiobjective optimization is used to compare the individuals in the population. Effects of the self-adaptive perturbation rate can control the frequency of parameter change. 24 benchmark tests chosen from CEC2006 have been conducted in our experiments. Experimental results indicate that our approach is effective and efficient. Compared with other state-of-the-art approach, SACABC performs better, or at least comparably, in terms of the quality of the final solutions and the convergence rate.

The rest of this paper is organized as follows: In Sect. 2 we will review the basic artificial bee colony. The problem method is presented in Sect. 3. Benchmark problems and the corresponding experimental results are given in Sect. 4. In the last section we conclude this paper and point out some future research directions.

2 Artificial bee colony algorithm

Artificial bee colony algorithm is an optimization algorithm inspired by how honeybee swarm finds food. In this algorithm, the model of the ABC algorithm consists of three kinds of bees: employed bees, onlooker bees, and scout bees. Employed bees are responsible for exploiting

the nectar sources explored before, sharing their information with onlookers within the hive. Then, onlooker bees receive information about the food sources and choose a food source to exploit depending on the information of nectar quality. The employee bee becomes scout whose food source has been abandoned [12]. Also half of the population is employed bees, and the other half of the population is onlooker bees in the basic ABC. The Matlab code of ABC algorithm can be downloaded from http://mf.erciyes.edu.tr/abc/.

In the initialization phase, it starts by associating all employed bees with randomly generated food solutions. The initial population of solutions is filled with SN number of randomly generated D dimensions. Let $X_i = \{x_{i1}, x_{i2}, \ldots, x_{iD}\}$ represent the ith food source in the population and SN be the number of food source which is equal to the number of the employed bees or onlooker bees. The constrained optimization is by the prescribed minimum and maximum bounds

$$\vec{X}_{\min} = \{x_{1,\min}, x_{2,\min}, \ldots, x_{D,\min}\}$$
$$\vec{X}_{\max} = \{x_{1,\max}, x_{2,\max}, \ldots, x_{D,\max}\}$$

Each food source is generated as follows:

$$x_{ij} = x_{j,\min} + (x_{j,\max} - x_{j,\min}) \times r \tag{2}$$

where $i \in \{1, 2, \ldots, \mathrm{SN}\}$ and $j \in \{1, 2, \ldots, D\}$ are randomly chosen indexes, r is a uniform random number in the range [0,1], and $x_{j,\min}$ and $x_{j,\max}$ are lower and upper bounds for the dimension j, respectively.

After initialization, this phase is the employed bees' phase; an employed bee can change its position x_{ij} in its memory depending on the local information to find a neighboring food source and evaluates the profitability of the nectar amount of the new food source v_{ij}. If the employed bee finds a better nectar source, it memorizes the new nectar source to use it instead of the old one. Then each employed bee x_{ij} generates a new food source v_{ij} in the neighborhood of its present position as follows:

$$v_{ij} = x_{ij} + \varphi_{ij}(x_{ij} - x_{kj}) \tag{3}$$

where $k = \mathrm{int}(\mathrm{rand} \times \mathrm{SN}) + 1$, $\varphi_{ij} = (\mathrm{rand} - 0.5) \times 2$ is a uniformly distributed real random number within the range $[-1, 1]$, $i \in \{1, 2, \ldots, \mathrm{SN}\}$, $k \in \{1, 2, \ldots, \mathrm{SN}\}$ and $k \neq i$, and $j \in \{1, 2, \ldots, n\}$ are randomly chosen indexes. After producing the new solution v_i, it will be evaluated and compared to the x_i. If the objective fitness of v_i is smaller than the fitness of x_i, v_i is accepted as a new basic solution. Otherwise x_i would be obtained. If the generated parameter value is out of boundaries, it is shifted into the bounties.

When all employed bees finish this process, an onlooker bee can obtain the information of the food sources from all employed bees and choose a food source depending on the probability value associated with the food source, using the following expression:

$$p_i = 0.9 \times \frac{\mathrm{fitness}_i}{\max(\mathrm{fitness}_i)} + 0.1 \tag{4}$$

where

$$\mathrm{fitness}_i = \begin{cases} \frac{1}{f_i} & f_i \geq 0 \\ 1 + |f_i| & f_i < 0 \end{cases} \tag{5}$$

where $\mathrm{fitness}_i$ is the fitness value of the solution i. Obviously, when the maximal value of the food source decreases, the probability with the preferred source of an onlooker bee decreases proportionally. The onlooker bee produces a new source according to Eq. (4). A greedy selection is applied on the new and original food sources.

In scout bees' phase, if a position cannot be improved further through a determined number of the trails "limit," the food source will be abandoned. Its employed bee will become a scout and then will discover a food source randomly according to Eq. (2).

These three phases will recycle until the termination condition is met. Then the algorithm outputs the best food source.

3 Proposed method: SACABC

In this section, SACABC is proposed in detail. In order to further improve the convergence speed and supply more valuable information to balance the global search capability and local search capability of the ABC, the employee bee colony phase is as the global search model based on the feasible rule. The employed bee colony phase is as the global search model based on the feasible rule. The onlooker bee phase is as the local search model based on multiobjective optimization. The feasible rule is simplicity and flexibility, which makes the feasible rule can couple to any sort of selection mechanism. However, they are prone to cause premature convergence. For multiobjective optimization, the main idea of this method can convert constrain optimization to unconstrained multiobjective optimization. The method can maintain the good infeasible solution to avoid the algorithm into the optimal solution. Therefore, in this paper, the algorithm combines two constrain methods. In addition, inspired by the differential evolution algorithm, two new search mechanisms are proposed to enhance the search ability and maintain population diversity. In [17], compared with standard algorithm, modified rate is proposed to make the algorithm that can change many parameters to improve the convergence rate. Self-adaptive modified rate is used in this paper. The value of MR can change according to the record of recent successful update probability and uses them to guide the generation of new MR.

The general framework of SACABC is depicted as follows:

Procedure SACABC

Begin

Generation t = 1;

Initialized with random vector values, and initialize parameters NP, D;

Evaluate the fitness and the constraint of every individual $f(x_i)$ and $G(x_i)$, i = 1,…,NP

and find the best individual with the best objective value according to the feasible rule. ;

While (stopping criterion is not met)

Use the self-adaptive employ bee colony as the global search model based on feasible rule

Evaluate the fitness and the constraint for the bee colony

Use the self-adaptive onlooker bee colony as the global search model based on multiobjective optimization

Update the generation number t = t + 1

End while

End.

3.1 Employed bee model based on feasible rule

Inspired by the mutation scheme of differential evolution, in this section, we propose a random search mechanism to improve the original ABC algorithm. As we know, differential evolution is an evolutionary algorithm first introduced by Storn and Price [10]. The crucial idea behind DE is a scheme for producing trial vectors according to the manipulation of target vector and difference vector. Mutation is an operation that adds a vector differential to a population vector of individuals to generate new trail vector. Crossover operation is followed, and a new trial vector is generated. For each part of the trail vector, a random number is generated; if this is lower than the crossover CR by the user, then the individual of the mutation trail vector is used. Different kinds of strategies of DE have been proposed based on the target vector selected and the number of difference vectors used. The following is a mutation strategy frequently used in the literature:

DE/rand/1:

$$v_i = x_a + F(x_b - x_c) \tag{6}$$

where a, b, and c are mutually different random integer indices selected from {1,…, SN}. F is the scaling factor or amplification factor, is a positive real number. Based on DE algorithm and the property of ABC, we propose a novel search mechanism to improve ABC:

$$\begin{aligned} v_{ij} &= x_{aj} + \varphi_{ij}(x_{ij} - x_{bj}) \\ k &= \text{int}(\text{rand} \times (\text{SN} - 1)) + 1; \end{aligned} \tag{7}$$

where $\varphi_{ij} = (\text{rand} - 0.5) \times 2$ is a uniformly distributed real random number within the range $[-1, 1]$, $i \in \{1, 2, \ldots, \text{SN}\}$, $k \in \{1, 2, \ldots, \text{SN}\}$ and $k \neq i$, and $j \in \{1, 2, \ldots, n\}$ are randomly chosen indexes. The new search method can drive the new candidate solutions only around the random solutions of the previous iteration.

Akay and Karaboga [17] propose a modified ABC algorithm by controlling the frequency of perturbation. Inspired by this improved version, we also use a control parameter, that is, modification rate (MR), in IABC. In order to produce a candidate food position v_{ij} from the current memorized x_{ij}, MABC algorithm uses the following expression:

$$v_{ij} = \begin{cases} x_{aj} + \varphi_{ij}(x_{ij} - x_{bj}), & \text{if} \quad r_{ij} \leq \text{MR} \\ x_{ij} & \text{otherwise} \end{cases} \tag{8}$$

In order to make the employ bee colony to solve the constrained optimization problem, feasible rule is incorporated into the employed bee colony phase. When a solution x_1 is compared to a solution x_2, the feasible rule can be described as follows [24]:

1. If both solutions are feasible, the smaller value x_1 is better than the x_2.
2. x_1 is feasible, and x_2 is infeasible.
3. If both solutions are infeasible, the smaller degree of constraint violation x_1 is better than the x_2.

3.2 Onlooker bee search model–based multiobjective optimization

When all employed bees finish this process, an onlooker bee can choose a food source depending on the probability value p_i associated with the fitness value, using the following expression:

$$p_i = 0.9 \times \frac{\text{fitness}_i}{\max(\text{fitness}_i)} + 0.1 \tag{9}$$

where fitness_i is the fitness value of the solution i. Obviously, when the maximal value of the food source decreases, the probability with the preferred source of an onlooker bee decreases proportionally.

Inspired by the De/current-to-rand/1 mutation scheme of differential evolution, in this section, we propose another search mechanism to improve the original onlooker bee colony phase. The following is a mutation strategy frequently used in the literature:

DE/current-to-rand/1:

$$v_i = x_i + F(x_i - x_a) + F(x_b - x_c) \tag{10}$$

where a, b, and c are mutually different random integer indices selected from {1,..., SN}. F is the scaling factor or amplification factor, is a positive real number.Based on the property of two algorithms, we propose a novel search mechanism to improve ABC:

$$\begin{aligned} v_{ij} &= x_{aj} + \varphi_{ij}(x_{ij} - x_{bj}) + \varphi_{ij}(x_{cj} - x_{dj}) \\ k &= \text{int}(\text{rand} \times (\text{SN} - 1)) + 1; \end{aligned} \tag{11}$$

where $\varphi_{ij} = (\text{rand} - 0.5) \times 2$ is a uniformly distributed real random number within the range $[-1, 1]$, $i \in \{1, 2, \ldots, \text{SN}\}$, $k \in \{1, 2, \ldots, \text{SN}\}$ and $k \neq i$, and $j \in \{1, 2, \ldots, n\}$ are randomly chosen indexes. This new search mechanism can enhance the diversity of the population.The onlooker bee produces a new source from the old one in memory as follows:

$$v_{ij} = \begin{cases} v_{ij} = x_{aj} + \varphi_{ij}(x_{ij} - x_{bj}) + \varphi_{ij}(x_{cj} - x_{dj}), & \text{if } r_{ij} \leq \text{MR} \\ x_{ij} & \text{otherwise} \end{cases} \tag{12}$$

where $\phi_{ij} = (\text{rand} - 0.5) \times 2$ is a uniformly distributed real random number within the range $[-1, 1]$, $i \in \{1, 2, \ldots, \text{SN}\}$, $k \in \{1, 2, \ldots, \text{SN}\}$ and $k \neq i$, and $j \in \{1, 2, \ldots, n\}$ are randomly chosen indexes.

For the constrained optimization, the multiobjective optimization method is as the constraint handling mechanism. The method converts constrained optimization problem into two objective problems $\vec{f}(x) = (f(x), G(x))$. The degree of constraint violation $G(x)$ is as the new objective function. For the constrained optimization problem with two conflicting objective functions, its any two solutions have one of two possibilities: One dominates the other, or none dominates the other. Therefore, Pareto dominance is very important in multiobjective optimization problem and is adopted to compare the individuals. In particular, a feasible solution $\vec{f}(x_1)$ is said to dominate a feasible solution $\vec{f}(x_2)$, it can be denoted by $\vec{f}(x_1) \prec \vec{f}(x_2)$, if one of the following two conditions is satisfied:

$$\begin{array}{lllll} (1) & f(x_1) \leq f(x_2) & \text{and} & G(x_1) \leq G(x_2) \\ (2) & f(x_1) < f(x_2) & \text{and} & G(x_1) < G(x_2) \end{array} \tag{13}$$

And a feasible solution x is also said to be non-dominated, if there does not exist another x' such as $\vec{f}(x') \prec \vec{f}(x)$.

3.3 Self-adaptive modification rate (MR)

In this section, self-adaptive modification rate is proposed to generate better solutions by the knowledge of their successful values. The setting of MR_i is self-adaptive performed. At each generation G, the crossover rate MR_i of each individual is independently generated as

$$\text{MR}_i = \text{randn}_i(\text{MR}_m, 0.1) \tag{14}$$

where $\text{randn}_i(\text{MR}_m, 0.1)$ is a random number sampled from a Gaussian distribution accordingly with mean MR_m and standard deviation 0.1. $\text{MR}_{\text{success}}$ is the set of all successful update probabilities at the current generation, and the mean MR_m is initialized to be 0.83 and then updated at the end of each generation as:

$$\text{MR}_m = (1 - \delta) \cdot \text{MR}_m + \delta \cdot \text{meanpow}(\text{MR}_{\text{success}}) \tag{15}$$

where δ is a positive constant. The power mean is calculated as:

$$\text{meanpow}(\text{MR}_{\text{success}}) = \sum_{x \in \text{MR}_{\text{success}}} (x^n / |\text{MR}_{\text{success}}|)^{1/n} \tag{16}$$

where $|\text{MR}_{\text{success}}|$ is the cardinality of the set of $\text{MR}_{\text{success}}$. In this paper, n is 1.5 based on the experiment results.

3.4 Boundary constraints

Then, SACABC algorithm assumes that the whole population should be in an isolated and finite space. However, during the searching process, if there are some individuals that will move out of bounds of the space, the original algorithm stops them on the boundary. In other words, the nest will be assigned a boundary value. The disadvantage is that if there are too many individuals on the boundary, and especially when there exists some local minimum on the boundary, the algorithm will lose its population diversity to some extent. In order to tackle this problem, we propose the following repair rule:

$$x_i = \begin{cases} \min(U_i, 2 \times L_i - x_i) \\ \max(L_i, 2 \times U_i - x_i) \end{cases} \tag{17}$$

4 Experimental results

Numerical simulations were executed based on 24 standards benchmark functions to evaluate the effectiveness of the SACABC algorithm. These functions have been widely used in the literature [25]. The first 13 test functions chose from [25] are used. Since we do not make any modification in these functions, they are given in Table 1, and these first 13 problems are described in Appendix. As shown in Table 1, D is the number of variables of the problem. LI is the number of linear inequalities. LE denotes the number of linear equalities. NE is the nonlinear equalities. NI is the number of nonlinear inequality constraints. In Table 1, A is the number of active constraints. $f(x^*)$ is the objective function value of

Table 1 Benchmark problems

Problem	D	Type of prob	LI	NI	LE	NE	A	Optimal
g01	13	Quadratic	9	0	0	0	6	−15.000
g02	20	Nonlinear	1	1	0	0	1	−0.803619
g03	10	Nonlinear	0	0	0	1	1	−1.000
g04	5	Quadratic	0	6	0	0	2	−30,665.539
g05	4	Nonlinear	2	0	0	3	3	5,126.498
g06	2	Nonlinear	0	2	0	0	2	−6,961.814
g07	10	Quadratic	3	5	0	0	6	24.306
g08	2	Nonlinear	0	2	0	0	0	−0.095825
g09	7	Nonlinear	0	4	0	0	2	680.63
g11	2	Quadratic	0	0	0	1	1	0.75
g12	3	Quadratic	0	9^3	0	0	0	−1.000
g13	5	Nonlinear	0	0	1	2	3	0.053950

D dimension of the problem, *LI* linear inequality, *NI* nonlinear inequality, *NE* nonlinear quality; a more detailed description of these functions can be found in the literature

the best-known solution. It has been noted that test functions g02, g03, g08, and g12 are maximization problems. In this paper, we convert the maximization problems to the minimization problem with $-f(x)$.

The algorithm is coded in MATLAB 7.0, and experiments are made on a Pentium 3.0 GHz Processor with 1.0 GB of memory.

In order to evaluate the effectiveness and efficiency of SACABC, we have chosen a suitable set of values and have not made any effort in finding the better parameter settings. In this experiment, we set the number of individuals to be 55. (In our algorithm, the scout bee colony phase is not used.) The tolerance value ε for the equality constraints is set to 0.0001. The number of fitness function evaluations (FFEs) is equal to 240,000. The maximum number of fitness function evaluations can be considered as a convergence rate. For all test functions, the algorithms carry out 25 independent runs. Each run is terminated only when the maximum number of generations has been elapsed. The results of SACABC algorithm are compared with the results of other methods with respect to the relevant literature in order to make fair comparison including homomorphous mapping (HM) method [26], simple multimember evolution strategy (SMES) [27], genetic algorithm (GA) [27], particle swarm optimization [28], differential evolution [20], artificial bee colony algorithm [20], HCOEA [14], ATMES [29], and M_ABC [18].

4.1 Experimental setup

The statistical results of SACABC using the above parameter setting are listed in Table 2. Table 2 shows the known optimal solution for each test function and the best, mean, worst, and standard deviation of the objective function values derived from SACABC over 30 independent runs. From Table 2, SACABC algorithm is able to find the global minimum of the 12 problems (g01, g03, g04, g05, g06, g07, g08, g09, g10, g11, g12, and g13) using the 240,000 cycles. Then, SACABC was able to find close results to global optimum for test function g02.

In order to evaluate the effectiveness and efficiency of SACABC future, the solutions derived from SACABC are compared with those provided by the HM, SMES, GA, PSO, DE, and ABC. First, it should be noted that the results of HM were obtained after 1,400,000, and the results of PSO algorithm were obtained after 350,000, while those of DE, GA, ABC, and SMES were obtained after 240,000. Tables 3 and 4 show the results of the best and mean solutions of the investigated algorithms. Results of the HM, SMES, GA, PSO, DE, and ABC are reported in [20, 27–29]. According to the best solutions in Table 3, it can be seen that SACABC obtains better solutions than HM, SMES, GA, PSO, DE, and ABC. Moreover, with respect to the test functions (g05, g11) with equality constrains, objective function values that are better than the optimal values have been found, it is because equality constraints are transformed into inequality constraints and relaxed by using the parameter ε. The best results show the ability of an algorithm to find the optimal result, while the mean results and standard deviation values show the robustness of the algorithm. Table 4 shows the mean results of these algorithms. From Table 4, it can be seen that the SACABC algorithm reached better or equal mean results than other algorithms. Therefore, we can find the SACABC algorithm is more efficient than HM, SMES, GA, PSO, DE, and ABC.

4.2 Comparison with the HCOEA and ATMES

To verify the effectiveness of the proposed method, we compare our approach with HCOEA [14] and ATMES [29]. Wang et al. propose a hybrid constrained optimization EA (HCOEA), which combines multiobjective optimization with global and local search model. In the global search model, a niching genetic algorithm is proposed. In the local search model, a clustering partition of the population and multiparent crossover is proposed. ATMES, which combines simple evolutionary strategy (ES) and adaptive trade-off model, is proposed for constrained evolutionary optimization. The experimental results are listed in Tables 5 and 6. It is proven that HCOEA and ATMES give very high-quality results for 13 benchmark test function. In Tables 5 and 6, the better results between two algorithms are highlighted using boldface. The FEEs of all algorithms is 240,000; the results are restricted to the

Table 2 Statistical results obtained by SACABC algorithm for 13 test functions over 30 independent runs using 240,000

Problem	Optimal	Best	Mean	Worst	Std
g01	−15.000	−15.000	−15.000	−15.000	0
g02	−0.803619	−0.803618	−0.7886	−0.7688	0.0141
g03	−1.000	−1.0005	−1.0004	−0.9997	2.55086e−004
g04	−30,665.539	−30,665.539	−30,665.539	−30,665.539	0
g05	5,126.498	5,126.497	5,126.497	5,126.497	9.5869e−013
g06	−6,961.814	−6,961.814	−6,961.814	−6,961.814	1.9174e−012
g07	24.306	24.3062	24.3062	24.3064	2.5434e−007
g08	−0.095825	−0.095825	−0.095825	−0.095825	1.4628e−017
g09	680.63	680.630	680.630	680.630	1.1984e−013
g10	7,049.248	7,049.248	7,049.248	7,049.248	3.0316e−013
g11	0.75	0.7499	0.7499	0.7499	1.1702e−016
g12	−1.000	−1.000	−1.000	−1.000	0
g13	0.05394	0.05394	0.09242	0.438802	0.1217

Table 3 The best solution obtained by the HM, GA, SMES, PSO, DE, ABC, and SACABC algorithm for 13 test functions over 30 independent runs

Problem	Optimal	HM [17]	GA [18]	SMES [18]	PSO [19]	DE [20]	ABC [21]	SACABC
g01	−15.000	−14.7864	−14.440	−15.000	−15.000	−15.000	−15.000	**−15.000**
g02	−0.803619	−0.79953	−0.796231	−0.803601	−0.6699159	−0.472	−0.803598	**−0.803618**
g03	−1.000	−0.9997	−0.990	−1.000	−0.993930	−1.000	−1.000	**−1.0005**
g04	−30,665.539	−30,664.5	−30,626.053	−30,665.539	−30,665.539	−30,665.539	−30,665.539	**−30,665.539**
g05	5,126.498	–	–	5,126.599	5,126.484	5,126.484	5,126.484	**5,126.497**
g06	−6,961.814	−6,952.1	−6,952.472	−6,961.814	−6,161.814	−6,954.434	−6,961.814	**−6,961.814**
g07	24.306	24.620	31.097	24.327	24.370153	24.306	24.330	**24.3062**
g08	−0.095825	−0.095825	−0.095825	−0.095825	−0.095825	−0.095825	−0.095825	**−0.095825**
g09	680.63	680.91	685.994	680.632	680.630	680.630	680.634	**680.630**
g10	7,049.25	7,147.9	9,079.770	7,051.903	7,049.381	7,049.248	7.53.904	**7,049.248**
g11	0.75	0.75	0.750	0.75	0.749	0.752	0.750	**0.7499**
g12	−1.000	NA	−1.000	−1.000	−1.000	−1.000	−1.000	**−1.000**
g13	0.053950	NA	0.134057	0.053986	0.085655	0.385	0.760	**0.05394**

The bold values represent the best value

(−) means that no feasible solutions were found. *NA* not available

first thirteen test problems due to the fact that no results were found for the rest of them.

Compared with ATMES, the statistical values in Table 5 indicate that SACABC has the similar "best," "mean," and "worst" results for 11 test functions (g01, g03, g04, g05, g06, g08, g11, and g12). For test function g07, g09, and g10, SACABC can obtain the better "best," "mean," and "worst" results. As can be seen in Table 5, SACABC can find better "best" and "worst" results for g02. The "mean" of the function g02 is obtained by the ATMES. For the function g13, both algorithms can find the "best" value. However, the "mean" and "worst" values are obtained by the ATMES.

Compared with HCOEA, SACABC finds similar "best," "mean," and "worst" results for 11 test functions (g01, g04, g05, g06, g08, g09, g11, and g12). For test function g02, better "mean" and "worst" results are found by HCOEA. SACABC surpasses HCOEA for test function g03 (except the "worst" value), g05, g07, and g11 in terms of all performance metrics. HCOEA can obtain better solution than the SACABC for the test function g13.

Based on the above results, SACBAC shows a very competitive performance with that of AIMES and HCOEA which are the state-of-the-art methods in the filed of constrained optimization problem.

4.3 Comparison with M-ABC

In order to evaluate the effectiveness of the proposed algorithm further, we compare our results with the M-ABC

Table 4 The mean solution obtained by the HM, GA, SMES, PSO, DE, ABC, and SACABC algorithm for 13 test functions over 30 independent runs

Problem	Optimal	HM [17]	GA [18]	SMES [18]	PSO [19]	DE [20]	ABC [21]	SACABC
g01	−15.000	−14.7082	−14.236	−15.000	−14.710	−14.555	−15.000	**−15.000**
g02	−0.803619	−0.79671	−0.788588	−0.785238	−0.419960	−0.665	−0.792412	−0.7886
g03	−1.000	−0.9989	−0.976	−1.000	−0.764813	−1.000	−1.000	**−1.0004**
g04	−30,665.539	−30,655.3	−30,590.455	−30,665.539	−30,665.539	−30,665.539	−30,665.539	**−30,665.539**
g05	5,126.498	5,126.498	–	5,174.492	5,135.973	5,264.270	5,185.714	**5,126.497**
g06	−6,961.814	−6,961.814	−6,872.204	−6,961.284	−6,961.814	–	−6,961.813	**−6,961.814**
g07	24.306	24.826	34.980	24.475	32.407	24.310	24.473	**24.3062**
g08	−0.095825	−0.0891568	−0.095799	−0.095825	−0.095825	−0.095825	−0.095825	**−0.095825**
g09	680.63	681.16	692.064	680.643	680.630	680.630	680.640	**680.630**
g10	7,049.25	8,163.6	10,003.225	7,253.047	7,205.5	7,147.334	7,224.407	**7,049.248**
g11	0.75	0.75	0.75	0.75	0.749	0.901	0.750	**0.7499**
g12	−1.000	NA	−1.000	−1.000	−0.998875	−1.000	−1.000	**−1.000**
g13	0.053950	NA	–	0.166385	0.569358	0.872	0.968	0.09242

The bold values represent the best value

(−) means that no feasible solutions were found. *NA* not available

Table 5 Comparing SACABC with respect to ATMES on 13 benchmark test functions

Function	Optimal	Best result		Mean result		Worst result	
		SACABC	ATMES	SACABC	ATMES	SACABC	ATMES
g01	−15.000	**−15.000**	**−15.000**	**−15.000**	−15.000	**−15.000**	−15.000
g02	−0.803619	**−0.803618**	−0.803388	−0.7886	−0.790148	−0.7688	−0.756986
g03	−1.000	**−1.0005**	−1.000	**−1.0004**	−1.000	−0.9997	−1.000
g04	−30,665.539	**−30,665.539**	−30,665.539	**−30,665.539**	−30,665.539	**−30,665.539**	−30,665.539
g05	5,126.498	**5,126.497**	5,126.498	**5,126.497**	5,127.648	**5,126.497**	5,135.256
g06	−6,961.814	**−6,961.814**	−6,961.814	**−6,961.814**	−6,961.814	**−6,961.814**	−6,961.814
g07	24.306	**24.3062**	24.306	**24.3062**	24.316	**24.3064**	24.359
g08	−0.095825	**−0.095825**	−0.095825	**−0.095825**	−0.095825	**−0.095825**	−0.095825
g09	680.63	**680.630**	680.630	**680.630**	680.639	**680.630**	680.673
g10	7,049.25	**7,049.248**	7,052.253	**7,049.248**	7,250.437	**7,049.248**	7,560.224
g11	0.75	**0.7499**	0.75	**0.7499**	0.75	**0.7499**	0.75
g12	−1.000	**−1.000**	−1.000	**−1.000**	−1.000	**−1.000**	−0.994
g13	0.053950	**0.05394**	0.05394	0.09242	0.053959	0.438802	0.053999

algorithm [18]. 24 benchmark test functions in the CEC2006 are used, which includes 13 benchmark test function in the previous experiments. For M-ABC algorithm, four modifications related with the selection mechanism, the scout bee operator, and the equality and boundary constrains are made to the algorithm with the aim of modifying the behavior in a search space. In our experiments, the parameter setting is the same as the previous one. The best, mean, and worst of the objective value for 30 runs are reported in Table 7. For the function g20 and g22, the M-ABC cannot find the feasible solutions in any runs. Our algorithm cannot find the feasible solution. For the functions g01, g04, g06, g08, g11, g12, g16, and g24, M-ABC and SACABC obtained the same statistical values. For the function g02, the best value can be found by SACABC, but the mean and worst values of the function are obtained by the M-ABC. For the function g03, SACABC gives the best and mean value. M-ABC gives the best "Worst" value. SACABC provided a "better" best, mean, and "worst" value with respect to M_ABC in test functions g05, g07, g09, g10, g13, g14, g15, g17, g18, g19, g21, and g23. The overall results show that SACABC has the ability to provide high-quality solutions with respect to the M-ABC in most of the test functions in this paper.

Table 6 Comparing SACABC with respect to HCOEA on 12 benchmark test functions

Function	Optimal	Best result		Mean result		Worst result	
		SACABC	HCOEA	SACABC	HCOEA	SACABC	HCOEA
g01	−15.000	**−15.000**	−15.000	**−15.000**	−15.000	**−15.000**	−15.000
g02	−0.803619	**−0.803618**	−0.803241	−0.7886	−0.801258	−0.7688	−0.792363
g03	−1.000	**−1.0005**	−1.000	**−1.0004**	−1.000	−0.9997	−1.000
g04	−30,665.539	**−30,665.539**	−30,665.539	**−30,665.539**	−30,665.539	**−30,665.539**	−30,665.539
g05	5,126.498	**5,126.497**	5,126.498	**5,126.497**	5,126.498	**5,126.497**	5,126.498
g06	−6,961.814	**−6,961.814**	−6,961.814	**−6,961.814**	−6,961.814	**−6,961.814**	−6,961.814
g07	24.306	**24.3062**	24.306	**24.3062**	24.307	**24.3064**	24.309
g08	−0.095825	**−0.095825**	−0.095825	**−0.095825**	−0.095825	**−0.095825**	−0.095825
g09	680.63	**680.630**	680.630	**680.630**	680.630	**680.630**	680.630
g10	7,049.25	**7,049.248**	7,049.287	**7,049.248**	7,049.525	**7,049.248**	7.49.984
g11	0.75	**0.7499**	0.750	**0.7499**	0.750	**0.7499**	0.750
g12	−1.000	**−1.000**	−1.000	**−1.000**	−1.000	**−1.000**	−1.000
g13	0.053950	**0.05394**	0.0539498	0.09242	0.0539498	0.438802	0.0539498

Table 7 Comparing SACABC with respect to M-ABC on 24 benchmark test functions

Function	Optimal	Best result		Mean result		Worst result	
		SACABC	M-ABC	SACABC	M-ABC	SACABC	M-ABC
g01	−15.000	**−15.000**	**−15**	**−15.000**	−15	**−15.000**	−15
g02	−0.803619	**−0.803618**	−0.83615	−0.7886	−0.799336	−0.7688	−0.777438
g03	−1.000	**−1.0005**	−1.000	**−1.0004**	−1.000	−0.9997	−1.000
g04	−30,665.539	**−30,665.539**	**−30,665.539**	**−30,665.539**	−30,665.539	**−30,665.539**	−30,665.539
g05	5,126.498	**5,126.497**	5,126.736	**5,126.497**	5,178.139	**5,126.497**	5,317.196
g06	−6,961.814	**−6,961.814**	**−6,961.814**	**−6,961.814**	−6,961.814	**−6,961.814**	−6,961.814
g07	24.306	**24.3062**	24.315	**24.3062**	24.315	**24.3064**	24.854
g08	−0.095825	**−0.095825**	**−0.095825**	**−0.095825**	−0.095825	**−0.095825**	−0.095825
g09	680.63	**680.630**	680.632	**680.630**	680.647	**680.630**	680.691
g10	7,049.25	**7,049.248**	7,051.706	**7,049.248**	7,233.882	**7,049.248**	7,473.109
g11	0.75	**0.7499**	0.75	**0.7499**	0.75	**0.7499**	0.75
g12	−1.000	**−1.000**	**−1.00**	**−1.000**	**−1.000**	**−1.000**	−1.000
g13	0.053950	**0.05394**	0.053985	**0.09242**	0.158552	**0.438802**	0.442905
g14	−47.765	−47.7648	−47.641	**−47.7648**	−47.641	**−47.7648**	−46.537
g15	961.715	961.715	961.715	**961.715**	961.715	**961.715**	961.793
g16	−1.905	−1.905	−1.905	**−1.905**	**−1.905**	**−1.905**	−1.905
g17	8,853.540	8,853.534	8,866.618	**8,853.631**	8,987.459	**8,854.504**	9,165.219
g18	−0.866025	−0.866025	−0.866606	**−0.866025**	−0.7950187	**−0.866025**	−0.672216
g19	32.656	32.6556	33.285	**32.6560**	34.267	**32.6597**	35.746
g20	0.096700	–	–	–	–	–	–
g21	193.725	193.725	266.500	**233.0180**	306.609	**324.702**	329.960
g22	236.431	–	–	–	–	–	–
g23	−400.055	−400.055	−159.739	**−400.055**	−35.272	**−400.055**	109.010
g24	−5.508	−5.508	−5.508	**−5.508**	**−5.508**	**−5.508**	−5.508

The bold values represent the best value

5 Conclusions

In this paper, we propose a constrained version of self-adaptive constrained artificial bee colony algorithm (SACABC) to solve the constraint optimization problems. The employed bee colony phase is as the global search model based on the feasible rule. The onlooker bee phase is as the local search model based on multiobjective optimization. Self-adaptive modification rate is proposed to make the algorithm that can change many parameters to improve the convergence rate. The value of MR can change according to the record of recent successful update probability and use them to guide the generation of new MR. The proposed SACABC and some state-of-the-art methods experiment on 24 benchmark test functions. Numerical simulations show that the proposed algorithm can obtain results better than the ones reported in the literature. It has been concluded that SACABC algorithm can be efficiently used for solving these problems due to its simplicity, reliability, and robustness.

Acknowledgments This work is supported by the National Natural Science Foundation of China under Grant No. 60803102, 60473042, 60573067.

Appendix

g01:

$$\text{Minimize} \quad f(x) = 5\sum_{i=1}^{4} x_i - 5\sum_{i=1}^{4} x_i^2 - \sum_{i=5}^{13} x_i$$

subject to

$$\begin{aligned}
g_1(x) &= 2x_1 + 2x_2 + x_{10} + x_{11} - 10 \le 0\\
g_2(x) &= 2x_1 + 2x_3 + x_{10} + x_{12} - 10 \le 0\\
g_3(x) &= 2x_2 + 2x_3 + x_{11} + x_{12} - 10 \le 0\\
g_4(x) &= -8x_1 + x_{10} \le 0\\
g_5(x) &= -8x_2 + x_{11} \le 0\\
g_6(x) &= -8x_3 + x_{12} \le 0\\
g_7(x) &= -2x_4 - x_5 + x_{10} \le 0\\
g_8(x) &= -2x_6 - x_7 + x_{11} \le 0\\
g_9(x) &= -2x_8 - x_9 + x_{12} \le 0
\end{aligned}$$

where bounds are $0 \le x_i \le 1$ $(i = 1, \ldots, 9, 13)$, $0 \le x_i \le 100$ $(i = 10, 11, 12)$. The global optimum is at $x^* = (1, 1, 1, 1, 1, 1, 1, 1, 1, 3, 3, 3, 1)$, $f(x^*) = -15$.

g02:

$$\text{Maximize} \quad f(x) = -\left|\frac{\sum_{i=1}^{n}\cos^4(x_i) - 2\prod_{i=1}^{n}\cos^2(x_i)}{\sqrt{\sum_{i=1}^{n} i x_i^2}}\right|$$

Subject to

$$g_1(x) = 0.75 - \prod_{i=1}^{n} x_i \le 0$$

$$g_2(x) = \sum_{i=1}^{n} x_i - 7.5n \le 0$$

where $n = 20$ and $0 \le x_i \le 10$ $(i = 1, \ldots, n)$. The known global maximum is at $x^* = 1/\sqrt{n}$ $(i = 1, \ldots, n)$, $f(x^*) = 0.803619$. g_1 is close to being active $(g_1 = -10^{-8})$.

g03:

$$\text{Maximize} \quad f(x) = (\sqrt{n})^n \prod_{i=1}^{n} x_i$$

Subject to

$$h(x) = \sum_{i=1}^{n} x_i^2 - 1 = 0$$

where $n = 10$ and $0 \le x_i \le 1$ $(i = 1, \ldots, n)$, the global maximum is at $x^* = 1/\sqrt{n}$ $(i = 1, \ldots, n)$ where $f(x^*) = 1$.

g04:

$$\begin{aligned}
\text{Minimize} \quad f(x) = {} & 5.3578547x_3^2 + 0.8356891x_1x_5\\
& + 37.293239x_1 - 40,792.141
\end{aligned}$$

Subject to

$$\begin{aligned}
g_1(x) &= 85.334407 + 0.0056858x_2x_5 + 0.0006262x_1x_4\\
&\quad - 0.0022053x_3x_5 - 92 \le 0\\
g_2(x) &= -85.334407 - 0.0056858x_2x_5 - 0.0006262x_1x_4\\
&\quad + 0.0022053x_3x_5 \le 0\\
g_3(x) &= 80.51249 + 0.0071317x_2x_5 + 0.0029955x_1x_2\\
&\quad + 0.0021813x_3^2 - 110 \le 0\\
g_4(x) &= -80.51249 - 0.0071317x_2x_5 - 0.0029955x_1x_2\\
&\quad + 0.0021813x_3^2 + 90 \le 0\\
g_5(x) &= 9.300961 + 0.004702x_3x_5 + 0.0012547x_1x_3\\
&\quad + 0.0019085x_3x_4 - 25 \le 0\\
g_6(x) &= -9.300961 - 0.004702x_3x_5 - 0.0012547x_1x_3\\
&\quad - 0.0019085x_3x_4 + 20 \le 0
\end{aligned}$$

where $78 \le x_1 \le 102$, $33 \le x_2 \le 45$, $27 \le x_i \le 45$ $(i = 3, 4, 5)$. The optimum solution is $x^* = (78, 33, 29.995256025682, 45, 36, 7758129005788)$, where $f(x^*) = -30,665.539$.

g05:

Minimize

$$f(x) = 3x_1 + 0.000001x_1^3 + 2x_2 + \left(\frac{0.000002}{3}\right)x_2^3$$

Subject to

$$\begin{aligned}
g_1(x) &= -x_4 + x_3 - 0.55 \le 0\\
g_2(x) &= -x_3 + x_4 - 0.55 \le 0\\
h_1(x) &= 1{,}000\sin(-x_3 - 0.25) + 1{,}000\sin(-x_4 - 025)\\
&\quad + 894.8 - x_1 = 0\\
h_2(x) &= 1{,}000\sin(x_3 - 0.25) + 1{,}000\sin(x_3 - x_4 - 025)\\
&\quad + 894.8 - x_2 = 0\\
h_3(x) &= 1{,}000\sin(x_4 - 0.25) + 1{,}000\sin(x_4 - x_3 - 025)\\
&\quad + 1{,}294.8 = 0
\end{aligned}$$

where $0 \le x_1 \le 1{,}200$, $0 \le x_2 \le 1{,}200$, $-0.55 \le x_3 \le 0.55$, and $-0.55 \le x_4 \le 0.55$. The best-known solution is $x^* = (679.9453, 1{,}026.067, 0.1188764, -0.3962336)$, where $f(x^*) = 5{,}126.498$.

g06:

Minimize $f(x) = (x_1 - 10)^3 + (x_2 - 20)^3$

Subject to

$$\begin{aligned}
g_1(x) &= -(x_1 - 5)^2 - (x_2 - 5)^2 + 100 \le 0\\
g_2(x) &= (x_1 - 6)^2 - (x_2 - 5)^2 - 82.81 \le 0
\end{aligned}$$

where $13 \le x_1 \le 100$ and $0 \le x_2 \le 100$. The optimum solution is $x^* = (14.095, 0.84296)$ where $f(x^*) = -6{,}961.81388$.

g07:

Minimize

$$\begin{aligned}
f(x) &= x_1^2 + x_2^2 + x_1x_2 - 14x_1 - 16x_2 + (x_3 - 10)^2\\
&\quad + 4(x_4 - 5)^2 + (x_5 - 3)^2 + 2(x_6 - 1)^2 + 5x_7^2\\
&\quad + 7(x_8 - 11)^2 + 2(x_9 - 10)^2 + (x_{10} - 7)^2 + 45
\end{aligned}$$

Subject to

$$\begin{aligned}
g_1(x) &= -105 + 4x_1 + 5x_2 - 3x_7 + 9x_8 \le 0\\
g_2(x) &= 10x_1 - 8x_2 - 17x_7 + 2x_8 \le 0\\
g_3(x) &= -8x_1 + 2x_2 + 5x_9 - 2x_{10} - 12 \le 0\\
g_4(x) &= 3x_1^2 + 4(x_2 - 3)^2 + 2x_3^2 - 7x_4 - 120 \le 0\\
g_5(x) &= 5(x_1 - 2)^2 + 8x_2 + (x_3 - 6)^2 - 2x_4 - 40 \le 0\\
g_6(x) &= x_1^2 + 2(x_2 - 2)^2 - 2x_1x_2 + 14x_5 - 6x_6 \le 0\\
g_7(x) &= 0.5(x_1 - 8)^2 + 2(x_2 - 4)^2 + 3x_5^2 - x_6 - 30 \le 0\\
g_8(x) &= -3x_1 + 6x_2 + 12(x_9 - 8)^2 - 7x_{10} \le 0
\end{aligned}$$

where $-10 \le x_1 \le 10$ $(i = 1, \ldots, 10)$. The global optimum is $x^* = (2.171996, 2.363683, 8.773926, 5.095984, 0.9906548, 1.430574, 1.321644, 9.828726, 8.280092, 8.375927)$, where $f(x^*) = 24.3062091$.

g08:

Minimize $f(x) = \dfrac{\sin^3(2\pi x_1)\sin(2\pi x_2)}{x_1^3(x_1 + x_2)}$

Subject to

$$\begin{aligned}
g_1(x) &= x_1^2 - x_2 + 1 \le 0\\
g_2(x) &= 1 - x_1 + (x_2 - 4)^2 \le 0
\end{aligned}$$

where $0 \le x_1 \le 10$ $(i = 1, 2)$. The optimum solution is located at $x^* = (1.2279713, 4.2453733)$, where $f(x^*) = 0.0095825$.

g09:

Minimize

$$\begin{aligned}
f(x) &= (x_1 - 10)^2 + 5(x_2 - 12)^2 + x_3^4 + 3(x_4 - 11)^2\\
&\quad + 10x_5^6 + 7x_6^2 + x_7^4 - 4x_6x_7 - 10x_6 - 8x_7
\end{aligned}$$

Subject to

$$\begin{aligned}
g_1(x) &= -127 + 2x_1^2 + 3x_2^4 + x_3 + 4x_4^2 + 5x_5 \le 0\\
g_2(x) &= -282 + 7x_1 + 3x_2 + 10x_2^3 + x_4 - x_5 \le 0\\
g_3(x) &= -196 + 23x_1 + x_2^2 + 6x_6^2 - 8x_7 \le 0\\
g_4(x) &= 4x_1^2 + x_2^2 - 3x_1x_2 + 2x_3^2 + 5x_6 - 11x_7 \le 0
\end{aligned}$$

where $-10 \le x_1 \le 10$, $i = 1, \ldots, 7$. The global optimum is $x^* = (2.330499, 1.951372, -0.4775414, 4.365726, -0.6244870, 1.038131, 1.594227)$, where $f(x^*) = 680.6300573$.

g11:

Minimize $f(x) = x_1^2 + (x_2 - 1)^2$

Subject to

$$h(x) = x_2 - x_1^2 = 0$$

where $-1 \le x_1 \le 1$, $-1 \le x_1 \le 1$. The optimum solution is $x^* = (\pm 1/\sqrt{(2)}, 1/2)$, where $f(x^*) = 0.75$.

g12:

Minimize

$$f(x) = \frac{100 - (x_1 - 5)^2 - (x_2 - 5)^2 - (x_3 - 5)^2}{100}$$

Subject to

$$g_1(x) = (x_i - p)^2 + (x_2 - q)^2 + (x_3 - r)^2 - 0.0625 \le 0$$

where $0 \le x_1 \le 10$ $(i = 1, 2, 3)$ and $p, q, r = 1, \ldots, 9$. The global optimum is located at $x^* = (5, 5, 5)$, where $f(x^*) = 1$.

g13:

Minimize $f(x) = e^{x_1 x_2 x_3 x_4 x_5}$

Subject to

$$h_1(x) = x_1^2 + x_2^2 + x_3^2 + x_4^2 + x_5^2 = 0$$
$$h_2(x) = x_2 x_3 - 5x_4 x_5 = 0$$
$$h_3(x) = x_1^3 + x_2^3 + 1 = 0$$

where $-2.3 \leq x_1 \leq 2.3, i = 1, 2, -3.2 \leq x_1 \leq 3.2, i = 3, 4, 5$. The global optimum is $x^* = (-1.717143, 1.5957091, 1.8272, -0.736413, -0.763645)$, where $f(x^*) = 0.0539498$.

References

1. Leticia CC, Susana CE, Carlos ACC (2011) Solving constrained optimization problems with a hybrid particle swarm optimization algorithm. Eng Optim 43(8):843–866
2. Victoria SA, Susana CE, Carlos ACC (2010) A modified version of a T-cell algorithm for constrained optimization problems. Int J Numer Meth Eng 84(3):351–378
3. Guillermo L, Carlos ACC (2009) Boundary search for constrained numerical optimization problems with an algorithm inspired on the ant colony metaphor. IEEE Trans Evol Comput 13(2):350–368
4. Suman B (2004) Study of simulated annealing based algorithms for multiobjective optimization of a constrained problem. Comput Chem Eng 8:1849–1871
5. Horn J, Nafpliotis N, Goldberg DE (1994) A niched Pareto genetic algorithm for multiobjective optimization. Evol Comput 1:82–87
6. Martí L, García J, Berlangaa A, Coello Coellob CA, Molin JM (2011) MB-GNG: addressing drawbacks in multi-objective optimization estimation of distribution algorithms. Oper Res Lett 39(2):150–154
7. Clerc M, Kennedy J (2002) The particle swarm-explosion, stability, and convergence in a multidimensional complex space. IEEE Trans Evol Comput 6:58–73
8. Dorigo M, Maniezzo V, Colorni A (1996) The ant system: optimization by a colony of cooperating agents. IEEE Trans Syst Man Cybern Part B 26(1):29–41
9. Simon D (2008) Biogeography-based optimization. IEEE Trans Evol Comput 12(6):702–713
10. Storn R, Price K (1997) Differential evolution—a simple and efficient heuristic for global optimization over continuous space. J Global Optim 11:341–359
11. Noman N, Iba H (2008) Accelerating differential evolution using an adaptive local search. IEEE Trans Evol Comput 12(1):107–125
12. Karaboga D, Basturk B (2008) On the performance of artificial bee colony (ABC) algorithm. Appl Soft Comput 8(1):687–697
13. Cai ZX, Wang Y (2006) A multiobjective optimization-based evolutionary algorithm for constrained optimization. IEEE Trans Evol Comput 10(6):658–675
14. Wang Y, Cai ZX, Guo GQ, Zhou YR (2007) Multiobjective optimization and hybrid evolutionary algorithm to solve constrained optimization problems. IEEE Trans Syst Man Cybern B Cybern 37(3):560–575
15. Karaboga D, Gorkemli B, Ozturk C, Karaboga N (2012) A comprehensive survey: artificial bee colony (ABC) algorithm and applications. Artif Intell Rev. doi:10.1007/s10462-012-9328-0
16. Karaboga D, Basturk B (2007) A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm. J Global Optim 39(3):459–471
17. Akay B, Karaboga D (2012) A modified artificial bee colony algorithm for real-parameter optimization. Inf Sci 192(1):120–142
18. Mezura-Montes E, Cetina-Domínguez O (2012) Empirical analysis of a modified artificial bee colony for constrained numerical optimization. Appl Math Comput. doi:10.1016/j.amc.2012.04.057
19. Mezura-Montes E, Velez-Koeppel RE (2010) Elitist artificial bee colony for constrained real-parameter optimization. IEEE Congr Evol Comput 7:1–8
20. Karaboga D, Akay B (2011) A modified artificial bee colony (ABC) algorithm for constrained optimization problems. Appl Soft Comput 11(3):3021–3031
21. Akay B, Karaboga D (2012) Artificial bee colony algorithm for large-scale problems and engineering design optimization. J Intell Manuf. doi:10.1007/s10845-010-0393-4
22. Brajevic I, Tuba M (2012) An upgraded artificial bee colony (ABC) algorithm for constrained optimization problems. J Intell Manuf. doi:10.1007/s10845-011-0621-6
23. Wang Y, Cai ZX (2009) A hybrid multi-swarm particle swarm optimization to solve constrained optimization problems. Front Comput Sci China 3(1):38–52
24. Deb K (2000) An efficient constraint handling method for genetic algorithms. Comput Methods Appl Mech Eng 186(2–4):311–338
25. Runarsson TP, Yao X (2000) Stochastic ranking for constrained evolutionary optimization. IEEE Trans Evol Comput 4(3):284–294
26. Koziel S, Michalewicz Z (1999) Evolutionary algorithms, homomorphous mappings, and constrained parameter optimization. Evol Comput 7(1):19–44
27. Ezura-Montes E, Coello Coello CA (2003) A simple multimembered evolution strategy to solve constrained optimization problems. Technical report EVOCINV-04–2003, Evolutionary Computation Group at CINVESTAV, Sección de Computación, Departamento de Ingeniería Eléctrica, CINVESTAV-IPN, México D.F., México. Available in the Constraint Handling Techniques in Evolutionary Algorithms Repository at http://www.cs.cinvestav.mx/~constraint/
28. Muñoz-Zavala A, Hernández AA, Diharce ERV (2005), Constrained optimization via particle evolutionary swarm optimization algorithm (PESO). In: Beyer H-G, O'Reilly U-M, Arnold DV, Banzhaf W, Blum C, Bonabeau EW, Cant Paz E, Dasgupta D, Deb K, Foster JA, de Jong ED, Lipson H, Llora X, Mancoridis S, Pelikan M, Raidl GR, Soule T, Tyrrell A, Watson J-P, Zitzler E (eds) Proceedings of the genetic and evolutionary computation conference (GECCO'2005), June, vol 1, ACM Press, New York, Washington, DC, USA, pp 209–216. ISBN:1–59593-010–8
29. Wang Y, Cai ZX, Zhou YR, Zeng W (2008) An adaptive tradeoff model for constrained evolutionary optimization. IEEE Trans Evol Comput 12(1):80–92